Multivariate Analysis of Stock Prices

Group 11

Abhijit Krishna Menon   Manya Raman

December 8, 2019

CONTENTS

1 Background and Introduction 2
  1.1 The Problem 2
  1.2 Goal of the study 2
  1.3 Possible Solution 2
2 Data Exploration and Visualization 3
3 Data Preparation and Pre-processing 5
  3.1 Stock Data 6
  3.2 Company News Data 6
  3.3 Reddit Data 6
4 Data Mining Techniques and Implementation 7
  4.1 Sentiment Analysis 7
  4.2 Linear Regression Techniques 7
  4.3 Neural Networks 9
  4.4 Recurrent Neural Network - Long Short Term Memory 11
5 Performance Evaluation 14
6 Discussion and Recommendation 16
7 Summary 16
8 Appendix: R Code for use case study 17
  8.1 Data Collection and Pre Processing. 17
  8.2 Data Visualization 21
  8.3 Modelling 24

[1] *Department of Data Analytics Engineering, Northeastern University, Boston, United States*

1 BACKGROUND AND INTRODUCTION

1.1 The Problem

The stock market is not a clear cut equation and it depends on a lot of factors and not just numbers.In this world dominated by fake news and headlines, a lot of our economic decisions are made through what we see taking place around us. One such example is that negative news will normally cause individuals to sell stocks. A single headline of a company does not only change the stocks of that only but also affects the other companies related to it.What we are trying to achieve via this project is to determine how much does a single news article affect the stock of a particular company stocks.

1.2 Goal of the study

The stock market prediction has been an active area of research for quite a while. However, building a model that takes into consideration every factor is still a challenging problem. Apart from historical prices, the current stock market is affected by news articles about the company, general news and many other micro economic and macroeconomic factors. There are several models out there to predict the stocks based on general news elements or historical prices but none have taken into consideration all company specific news or all of these factors together.

In this project, our goal is to analyse how much of a role do these factors play while attempting to predict the next day's stock price.Thus, making a model with checking all possible computations and permutations that improves it's accuracy and delivers an efficient model.

1.3 Possible Solution

Our project will focus on two tracks that join into one rail at the end.

The first track will be to build a sentiment analysis model using the common news articles and the company specific news articles.

The second track is to utilize the numerical data available to us about the stocks and linearly model this data with our final target.

Finally, using the sentiment generated from the first track we would like to see if the sentiment scores generated actually influence our final output as compared to when we do not utilize it for predictions.

1.3.1 *Github Repository*

The link to our github repository including the code and other details of our project is https://github.com/akmenon1996/Multivariate-Stock-Market-Analysis

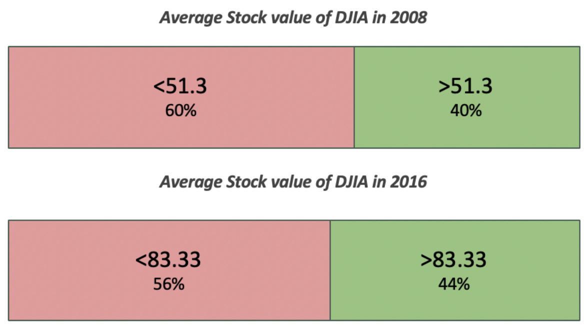

Figure 1: Summary of Average Stock Value change in 8 years.

2 DATA EXPLORATION AND VISUALIZATION

The stock data collected from Yahoo API of DJIA(Dow Jones Industrial Average) companies for 8 years has data missing on weekends data which is explained in next section.The news data collected from all the resources was filtered only for Business section to get relevant news only and it is not present for every day as there is no news of every company everyday thus the sentiment for that day is taken as zero.

After cleaning and transforming the data to a proper format, next step was to discover meaningful relations and summarize the main characteristics of the data set. For this purpose we performed a detailed Exploratory Data Analysis(EDA) on our data.

First we analyzed the average stock value in 2008, that was 51.30 and the DJIA companies above average were 40%. But in 2016 the metric changed and the average stock value increased becoming 83.33. But the interesting part we analyzed was that the companies above average are now different than some of those that were in 2008.

To understand this increase in percentage and change in companies stock above average we did an in-depth analysis country-wise. Therefore, figure 2 and figure 3 shows the change in stock value of 30 companies, from where we can see that companies like Apple, Disney, Home Depot etc. which were below average in 2008 are above average in 2015, and companies like Walmart, Procter & Gamble and Caterpillar Inc. which were above average in 2008 are now below average.

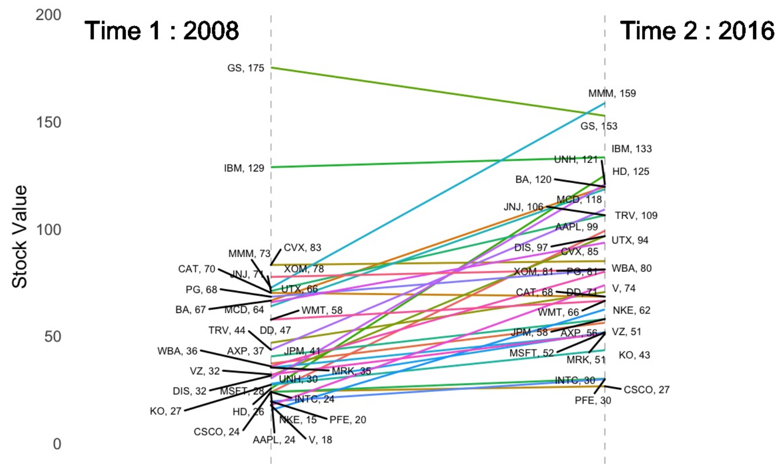

Figure 2: Slope Plot for change in stock value over time

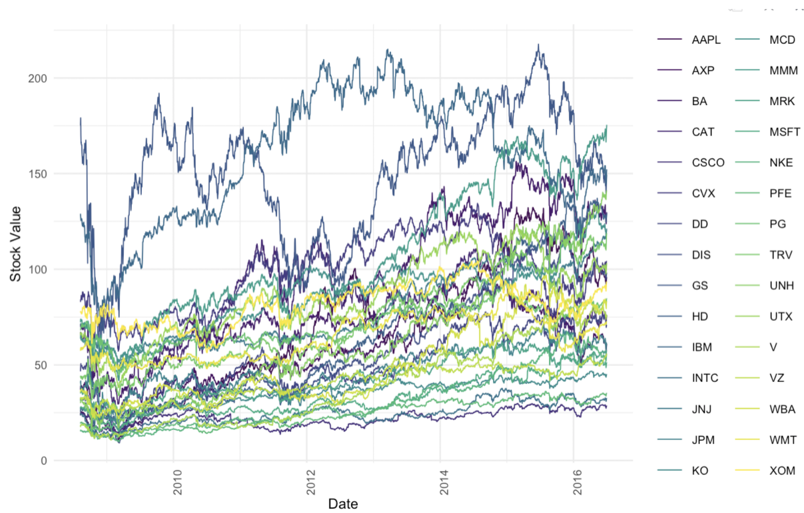

Figure 3: Line Plot for change in stock value over time

After visualization of stock data we started visualizing the correlation between different variables as seen from the correlation plot in figure 4.We analyzed that there is correlation between stock value and common sentiment thus we did an in-depth analysis.

Figure 4: Correlation Plot between input variables and target.

To visualize correlation between stock value and sentiment we first removed the trend from the stock data and lagged the stock value by 1 as today's news sentiment will affect the stock value of tomorrow. Figure 5 displays the zoomed version of Apple, IBM and Walgreens stacks and news sentiments respectively. We can see that from the figures that there is a directly proportional relationship between news sentiment and stock value.

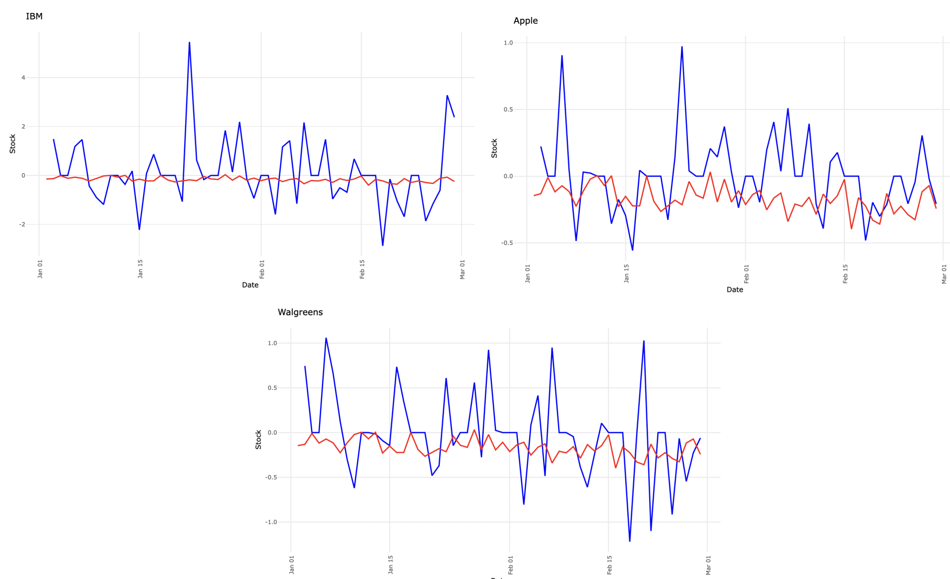

Figure 5: Sentiment(Red) and Stock(Blue) for IBM,Apple and Walgreens.

3 DATA PREPARATION AND PRE-PROCESSING

The data is collected from 3 different sources and pre -processed in different manner :

3.1 Stock Data

We have collected data of 30 companies over a period of 2006 to 2016 from Yahoo finance API. The attributes are: Opening price, Closing price, high and low.The stock data is absent for weekends and other holidays when the market is closed. In order to complete the data, we approximate the missing values using concave function as the stock data usually follows this function. So, if the DJIA value on a given day is x and the next available data point is y with n days missing in between, we approximate the missing data by estimating the first day after x to be (y+x)/2 and then following the same method recursively until all gaps are filled.

3.2 Company News Data

The data for 30 companies under Dow Jones Industrial Average (DJIA) is collected from the NY-Times API. The attributes are : created_time,snippet,headline, news desk and company name.The problem with data collection using the API was that it allows only 110 articles of any kind for a given date. We have mined data for over 6 years across 30 companies. We achieved this by writing a recursive function in R which takes into consideration the number of hits the API can make in a given minute and the number of pages it should look to as well as the year and company that we are looking at. After getting the data we filtered out the data on the condition of the section of news it originates from. The data was collected for every company individually thus we first joined all the companies files and created one with all.After that, a single string was formed from concatenating all the articles headlines for a single day company-wise as some companies have multiple headlines in a single day. Finally we removed all the punctuation's and byte characters thus getting a final cleaned and formatted version of company-wise news data.

3.3 Reddit Data

The data is collected from Kaggle. This data contains historical news headlines from Reddit World-News Channel. They are ranked by reddit users' votes, and only the top 25 headlines are considered for a single date. The first column is the "date", and the second column is the "news headlines". Hence, there are 25 lines for each date. For pre-processing, a single string was formed from concatenating all the articles headlines for a single day.

After all three files are collected they are combined into one to do all the visualizations and data mining. The news data was merged with Dow Jones Industrial Average (DJIA) companies stock index value on appropriate date (lag by 1 day) to get final data frame.

4 DATA MINING TECHNIQUES AND IMPLEMENTATION

4.1 Sentiment Analysis

For the calculation of the polarity score of the two news headlines of each company we used an in built sentiment score calculator that came as a part of the 'sentimentr' package on R. We pivoted our data so as to have two columns, one for the sentiment of the common news and one for the company specific news. We then calculated the sentiment for each of these news items and added them to the respective columns by day for each company. We had the sentiment for every day of the common news, however the company specific news was mostly 0s because not every day had a news item for every company, also we could not collect a lot of news data across multiple sources. If the sentiment for that day is NA then we decided to set the sentiment to 0 which essentially means the polarity for that day is neutral. Our assumption here is that a neutral polarity news items do not affect the stock price at all.

4.2 Linear Regression Techniques

For building the linear models, we first plotted scatter plots amongst all the variables with the target variable to visualize linearity(if any). From the scatter plots we notice that the only variable with a linear relation to the target value is the previous day's stock value, which was fairly expected. The volume of stocks traded has an exponential relation to the target after performing a log transformation and the news data does not have any heavy correlation. The scatter plots can be seen as below.

4.2.1 *Generalised Linear Model*

The formula that we used to build the initial model on R was the following.

```

Call:
lm(formula = target ~ stock_val + log(1/Volume) + sentiment_score +
    sentiment_common, data = train_stock_df)

Coefficients:
     (Intercept)          stock_val       log(1/Volume)     sentiment_score
    sentiment_common
         0.271537           0.999769            0.014899            0.116579
        0.002159


summary(linear_model)

Call:
lm(formula = target ~ stock_val + log(1/Volume) + sentiment_score +
    sentiment_common, data = train_stock_df)
```

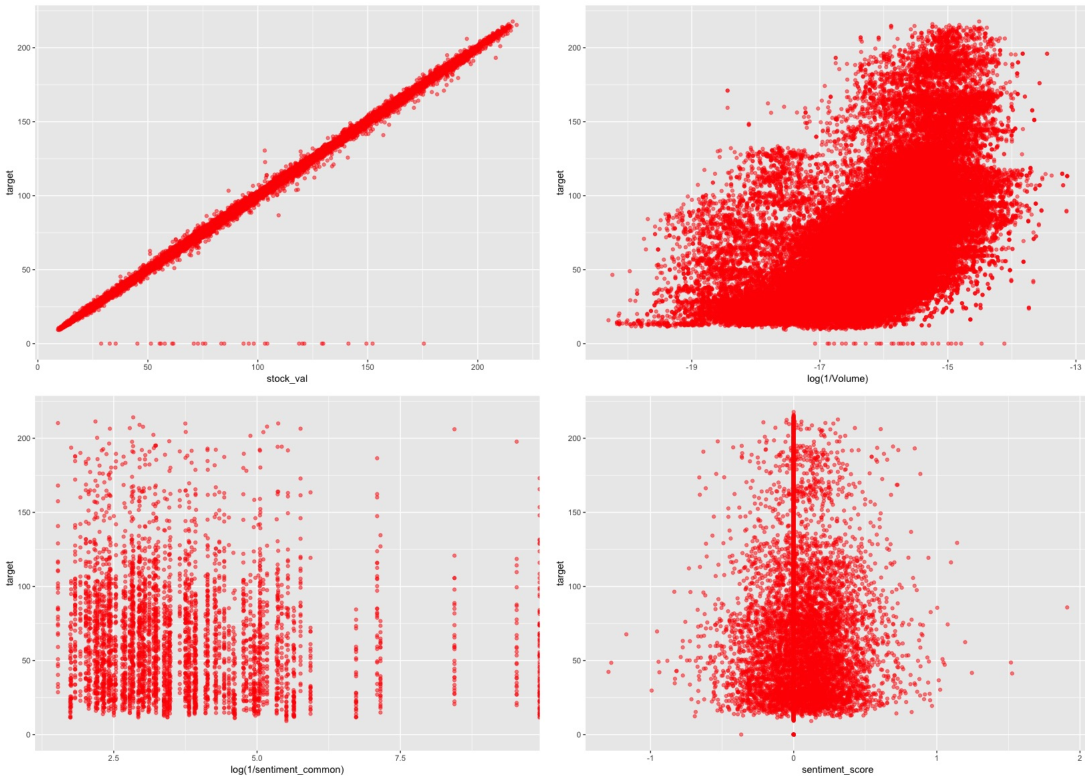

Figure 6: Linear relationship between input variables and target variable.

```

Residuals:
     Min       1Q   Median       3Q      Max
-22.7602  -0.1377  -0.0146   0.1661  27.3592

Coefficients:
                    Estimate Std. Error  t value Pr(>|t|)
(Intercept)        0.2715367  0.0643611    4.219 2.46e-05 ***
stock_val          0.9997692  0.0001047 9549.984  < 2e-16 ***
log(1/Volume)      0.0148994  0.0037255    3.999 6.36e-05 ***
sentiment_score    0.1165795  0.0434593    2.682  0.00731 **
sentiment_common   0.0021588  0.0228402    0.095  0.92470
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1
    1

Residual standard error: 0.7771 on 59155 degrees of freedom
Multiple R-squared:  0.9995,  Adjusted R-squared:  0.9995
F-statistic: 3.082e+07 on 4 and 59155 DF,  p-value: < 2.2e-16
```

From the summary of the linear model we see that the most important variables are:

- Stock Value
- 1/Volume

- Sentiment Score

On testing and generating our accuracy, we get a shocking correlation accuracy of 99%, however this is obviously not the right way to model the Time series data that we have with us. The reason being the high amount of trend, seasonality, and the high correlation between the he previous day's stock to the next day's affects the accuracy of the output. This is thus not the right model to be using for our data.

```
                 actuals      predicted
actuals      1.0000000       0.9963687
predicteds 0.9963687       1.0000000
```

4.2.2 *Support Vector Regressor*

```
SVR_model <- svm(target ~ stock_val+log(1/Volume)+sentiment_score+sentiment_
    common, data=train_stock_df)  # build linear regression model on full data
print(SVR_model)

Call:
svm(formula = target ~ stock_val + log(1/Volume) + sentiment_score + sentiment
    _common, data = train_stock_df)


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.25
    epsilon:  0.1


Number of Support Vectors:  810
```

Simillarly with the SVR we get a hiogh correlation accuracy of 99%. Again because of the earlier reasons, it is not the right model to use for this sort of data.

```
                actuals    predicted
actuals      1.0000000   0.9929886
predicteds 0.9929886   1.0000000
```

4.3 Neural Networks

Since the linear models were clearly not the way to go ahead with the data that we had with us, we decided to push our data through a bunch of neural networks and test the accuracy of the data predicted and the fit with the model.

For our experimentation we built different models with adding more variables at a time to see the difference in the prediction accuracy. This way, we can see the effect(if any) of the news variables on the final outcome.

The model used across all iterations of model bulding were constant. Just the variables inserted into the models variable.

For the base case, we tested 3 dense layers with the rectified linear unit(RELU) activation function. The loss was measured by the mean square error and the metrics we used to check accuracy is the mean absolute error.

```
            build_model <- function() {

  model <- keras_model_sequential() %>%
    layer_dense(units = 64, activation = "relu",
                input_shape = dim(train_data)[2]) %>%
    layer_dense(units = 64, activation = "relu") %>%
    layer_dense(units = 1)

  model %>% compile(
    loss = "mse",
    optimizer = optimizer_rmsprop(),
    metrics = list("mean_absolute_error")
  )

  model
}

model <- build_model()
model %>% summary()



Model: "sequential_7"
___________________________________________________________________________
  _
Layer (type)                                                        Output
  Shape                                                   Param #
===========================================================================
dense_21 (Dense)                                                    (None,
  64)                                                     256
___________________________________________________________________________
  _
dense_22 (Dense)                                                    (None,
  64)                                                     4160
___________________________________________________________________________
dense_23 (Dense)                                                    (None,
  1)                                                      65
===========================================================================
Total params: 4,481
Trainable params: 4,481
```

```
Non-trainable params: 0
________________________________________________________________________________
```

The accuracy of these sections will be discussed in the Performance and Evaluation Section.

4.4 Recurrent Neural Network - Long Short Term Memory

From the results of the linear models and the traditional neural network we realise that the other variables are just not being captured enough by the model, because of the stupendously high correlation between the previous day's data and the next day's data which is what was our target.

Thus it was necessary to build a different model, while still not going completely down the track of time series modelling. We decided to build an LSTM model, which is a kind of recurrent neural network.

What is a RNN or a recurrent Neural network? A brief explanation from one of our sources might be able to explain this better.

Recurrent Neural Networks

Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again. Your thoughts have persistence.

Traditional neural networks can't do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It's unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones.

Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist.

Now that we have an understanding of what an RNN is, we get a clearer picture of what role it plays for our data. Although we are trying to generate a linear relationship between our data points, it simply does not exist. This is why the previous data points are as important as the current one while making a decision.

The LSTM network allows us "remember" previous data points as a history and add that as an input variable while generating the output of the present data set.

```
              rmsprop = optimizer_rmsprop(lr=0.00001, rho=0.9, epsilon=1e-08)
model <- keras_model_sequential()

model %>%
    layer_lstm(units            = 100,
               input_shape      = c(dim(train_X)[2],dim(train_X)[3] ),
               return_sequences = TRUE,
```

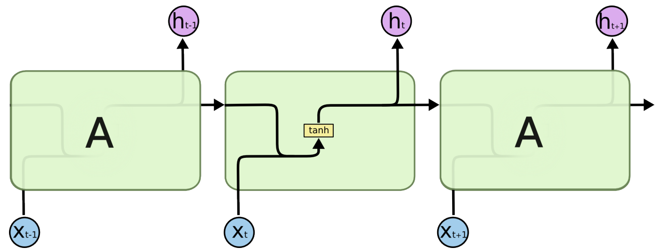

Figure 7: Network diagram of the LSTM model.

```
                 ) %>%
     layer_dropout(0.5)%>%
     layer_lstm(units             = 75,
                 return_sequences = FALSE,
                 ) %>%
   layer_dropout(0.5)%>%
   layer_flatten()%>%
   layer_dense(units = 64, activation = "relu") %>%
   layer_dropout(0.5)%>%
   layer_dense(units = 1)

model %>%
     compile(loss = 'mae', optimizer = rmsprop)

model


Model
Model: "sequential_3"
________________________________________________________________________________
Layer (type)                                                              Output
    Shape                                                     Param #
================================================================================
lstm_6 (LSTM)                                                             (None,
    1, 100)                                                   42000
________________________________________________________________________________
dropout_9 (Dropout)                                                       (None,
    1, 100)                                                   0
________________________________________________________________________________
lstm_7 (LSTM)                                                             (None,
    75)                                                       52800
________________________________________________________________________________
dropout_10 (Dropout)                                                      (None,
    75)                                                       0
________________________________________________________________________________
flatten_3 (Flatten)                                                       (None,
    75)                                                       0
          ______________________________________________________________________
    __________
```

```
dense_6 (Dense)                                                    (None,
    64)                                                   4864
________________________________________________________________________
dropout_11 (Dropout)                                               (None,
    64)                                                   0
________________________________________________________________________
dense_7 (Dense)                                                    (None,
    1)                                                    65
========================================================================
Total params: 99,729
Trainable params: 99,729
Non-trainable params: 0
________________________________________________________________________
```

For building the optimum model we used 2 LSTM layers, separated by a drop out layer to avoid overfitting. Followed a flatten layer, a dense layer and another dense layer of 1 that gives us the final output.

The input to the LSTM models is slightly different. The input is fed in as a 3d matrix. With the first dimensions being total number of records, the second being the numebr of time steps to be remembered and the 3rd dimensions as the input variables.

We trained this model over 50 epochs and a batch size of 10.

```
epochs = 50


history <- model %>% fit(
  train_X,
  train_y,
  epochs = epochs,
  batch_size = 10,
  validation_split = 0.2,
  verbose = 1,
  shuffle = FALSE
)
```

Using the LSTM, gave us satisfaction that the model is not over fit and is not overly dependant on a single variable. Using this model we made a few predictions, which we will cover over the next section.

5 PERFORMANCE EVALUATION

The model was split into 2 parts, the training data as well as the validation data in a 70-30 split. The performance evaluation of our various models are as follows:

- Linear Regression

  Mean Absolute Error of 4.82. Correlation Accuracy between Actual and Predicted values was 0.996.

- Support Vector Regression

  Mean Absolute Error of 3.22. Correlation Accuracy between Actual and Predicted values was 0.992.

- Neural Network - Only Stock Value and Volume.

  Mean absolute Error of 1.28.

- Neural Network - Stock Value, Volume and Sentiment of common news data.

  Mean absolute Error of 1.56.

- Neural Network - Only Sentiment columns

  Mean Absolute Error of 5.78

- Neural Network - All columns

  Mean Absolute Error of 1.0002.

- Recurrent Neural Network - LSTM (All columns)

  Mean Absolute Error of 0.45

  Thus we can see that the LSTM model proves to be the most accurate in terms of the MAE metric.

The company wise mean accuracy is also depicted below.

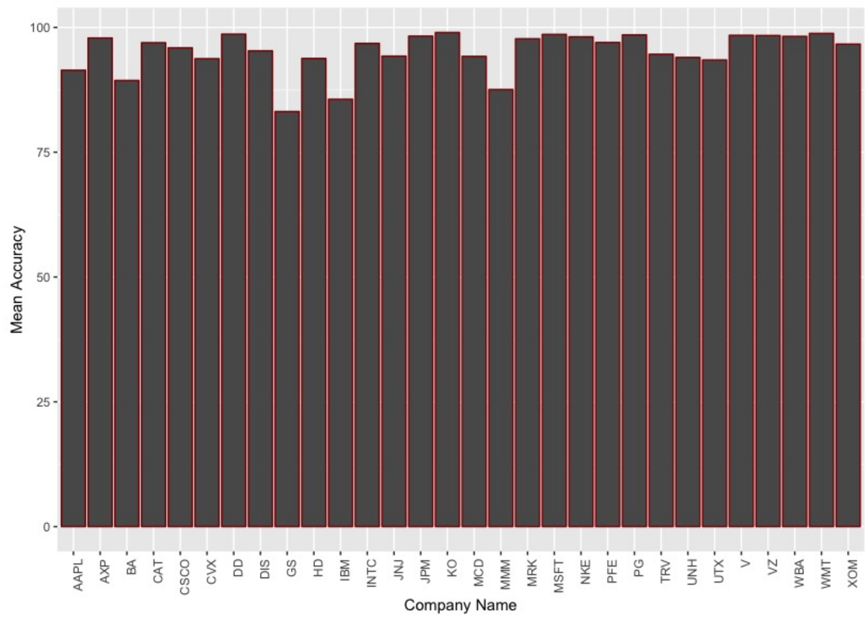

Figure 8: Company wise accuracy

The results of a few companies predictions are below. The red lines are the predicted values.

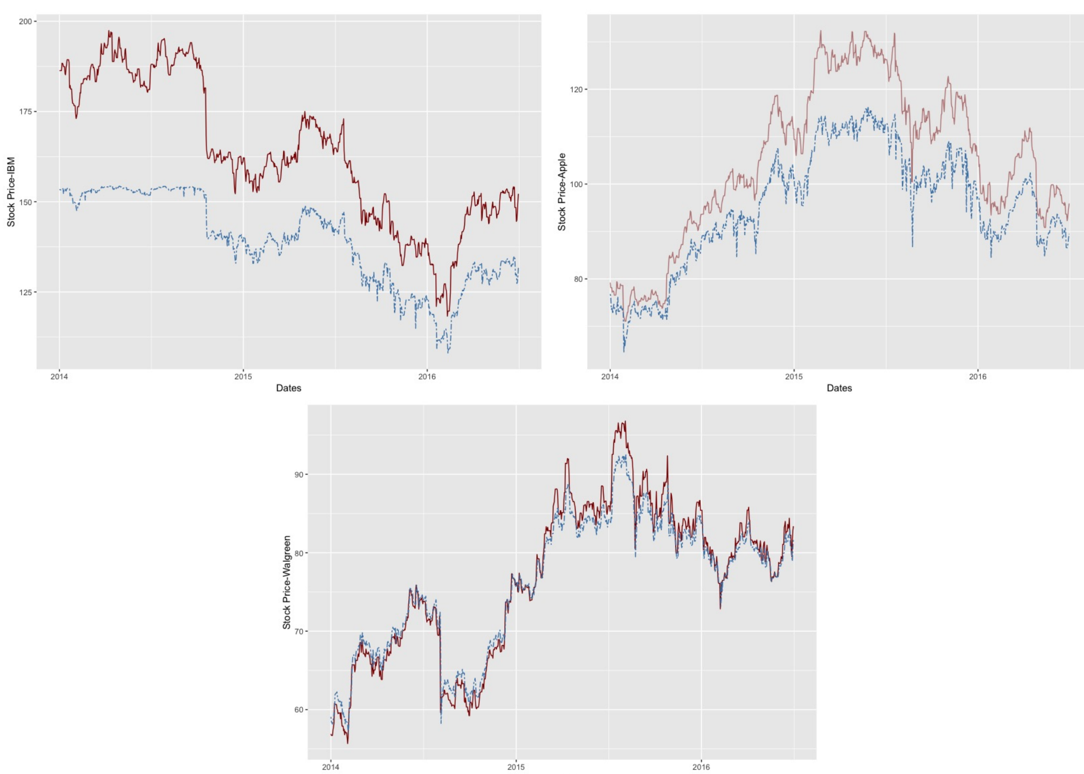

Figure 9: Predictions of data for IBM,Apple and Walgreens.

6 DISCUSSION AND RECOMMENDATION

Our primary issue with the data was that it is a time series model that we are trying to evaluate linearly. To get a better performanace and a better understanding of the data, we need to first work on a few time series techniques of removing trend, seasonality etc and work with a few time series models such as ARIMA etc. Next, we notice that from our data, the company news sentiment does not seem to be playing a massive role in the stock prices, however we know that, that is not true. Company news does indeed play a massive role in the stock prices. The problem with our data was that we were not able to collect too much of company news data for it to be viable from the New York times API. Solving this issue should have us gain a larger understanding on the effect that the news has on the stock prices.

7 SUMMARY

From our analysis, we see that sentiment does play a slight role on the price of the next day's stock, although not a whole lot. This is mainly because our news data set is not the most exhaustive resource there is. There is scope for a lot more improvement on the analysis of the affect of news sentiment on the stock prices.

Amongst the models we notice that the LSTM model performs considerable better than the other models. This does not come as a surprise since the LSTM model is the only model that takes into consideration the time series nature of this data set.

8 APPENDIX: R CODE FOR USE CASE STUDY

Our whole code can be found via the GIT HUB repository below. Some of the major chunks have been added below.

8.1 Data Collection and Pre Processing.

8.1.1 *Pre-processing of News data from NYTimes API*

````
---
title: "R Notebook"
output: html_notebook
---
```{r}
library(dplyr)
```

```{r}
library('googledrive')
api = "Nql4mIbzy44BETXG70DGesoOoXlnXeKH"
```



```{r}
if (!require("jsonlite")) install.packages("jsonlite")
library(jsonlite)
################################################
####            function - search news article with API ####
nytime = function (keyword,year) {
  searchQ = URLencode(keyword)
  url = paste('http://api.nytimes.com/svc/search/v2/articlesearch.json?q=',searchQ,
              '&begin_date=',year,'0101&end_date=',year,'1231&api-key=',api,sep="")
  #get the total number of search results
  initialsearch = fromJSON(url,flatten = T)
  maxPages = round((initialsearch$response$meta$hits / 10)-1)

  #try with the max page limit at 10
  maxPages = ifelse(maxPages >= 10, 10, maxPages)
  #creat a empty data frame
  df = data.frame(id=as.numeric(),created_time=character(),snippet=character(),
                  headline=character(),news_desk = character(),company_name = character())

  #save search results into data frame
````
````

```r
  r <- NULL
  attempt <- 1
  while( is.null(r) && attempt <= 3 ) {
    print(paste0("Try:",attempt))
    attempt <- attempt + 1
    Sys.sleep(10)
    try(
      for(i in 0:maxPages){
        #get the search results of each page
        nytSearch = fromJSON(paste0(url, "&page=", i), flatten = T)
        temp = data.frame(id=1:nrow(nytSearch$response$docs),
                          created_time = nytSearch$response$docs$pub_date,
                          snippet = nytSearch$response$docs$snippet,
                          headline = nytSearch$response$docs$headline.main,
                          news_desk = nytSearch$response$docs$news_desk,
                          company_name = keyword)
        df=rbind(df,temp)
        Sys.sleep(10)
        }
    )
  }
  return(df)
}
```

```{r}
company_list = c('Microsoft','Nike','Pfizer','Procter & Gamble',' The Travelers Companies','United Health Group','United Technologies','Verizon','Visa Inc.','Walmart','Walgreens Boots Alliance','The Walt Disney Company')
years = c(2008,2009,2010,2011,2012,2013,2014,2015)
```



```{r}
companies_df =  data.frame(id=as.numeric(),created_time=character(),snippet=character(),
                  headline=character(),news_desk = character(),company_name = character())
for (company in company_list){
  cat(paste0("Currently getting the data for ",company,"\n"),file = "Company_Data/data_gen.txt",append = TRUE)
  print(paste0("Currently getting the data for ",company))
  company_df =  data.frame(id=as.numeric(),created_time=character(),snippet=character(),
                  headline=character(),news_desk = character(),company_name = character())
  for(year in years){
    print(paste0("      ",year))
```

```
    cat(paste0("      Currently getting the data for the year ",year,"\n"),file
    = "Company_Data/data_gen.txt",append = TRUE)
    df = nytime(company,year)
    company_df <- rbind(company_df,df)
  }
  file1 = paste0("Company_Data/",company,".csv")
  print(file1)
  write.csv(company_df,file = file1)
  companies_df <- rbind(companies_df,company_df)
  cat(paste0("Succesfully got the data for ",company,"\n"),file = "Company_
    Data/data_gen.txt",append = TRUE)
}
write.csv(companies_df,file = "Company_Data/Final_Dataframe.csv")

```
```

8.1.2 *News data Preprocessing*

```
---
title: "News data Preprocessing"
output: html_document
---

```{r setup, include=FALSE}
library(readr)
library(plyr)
library(dplyr)
```

R Markdown

#Creating combined dataset by including all companies news data. NYTimes Data:

```{r concatenating_files}

file_list <- list.files()
for (file in file_list){

  # if the merged dataset doesn't exist, create it
  if (!exists("dataset")){
    dataset <- read.table(file, header=TRUE, sep=",")
  }

  # if the merged dataset does exist, append to it
  if (exists("dataset")){
    temp_dataset <-read.table(file, header=TRUE, sep=",")
    dataset<-rbind(dataset, temp_dataset)
    rm(temp_dataset)
  }
```

```
}
final_df <- dataset
rm(dataset)

```

```{r preprocessing data}

final_df_new<- final_df
final_df_new$created_time<-as.Date(final_df$created_time,format="%Y-%m-%d")

company_data <-final_df_new %>%
                    filter(news_desk == 'Business' & created_time >= '2008-08-08
    ' & created_time <= '2016-07-01')%>%
                    select(company_name,created_time,headline)


company_data<-ddply(company_data, .(company_name,created_time), summarise,
    headline=paste0(headline, collapse="; "))

levels(company_data$company_name)

write.csv(company_data, file = "/Users/manyaraman/Desktop/Comapny_Data.csv",
    row.names = FALSE)
```

#Preprocessing and formatting Reddit data :

```{r reddit_news_data}

Combined_news<-read_csv('Combined_News_DJIA.csv')
Combined_news<-Combined_news[,-2]

for(i in colnames(Combined_news[,-1])){
Combined_news[[i]]<-str_sub(Combined_news[[i]], 2)
  }

Combined_news_final <- unite(Combined_news, com_headlines, Top1:Top25, sep = "
    ;", remove = TRUE)

Reddit_news<-read_csv('RedditNews.csv')
Reddit_news<-ddply(Reddit_news, .(Date), summarise, News=paste0(News ,collapse
    ="; "))

Reddit_news$News<-str_replace_all(Reddit_news$News,c("b'" = "", "b\"" = ""))

write.csv(Reddit_news, file = "/Users/manyaraman/Desktop/Combined_news_final_
    reddit_wknd.csv", row.names = FALSE)
```
```
```

```

```

8.2 Data Visualization

```

---
title: "Project-visualization"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
library(ggplot2)
library(dplyr)
library(plyr)
library(readxl)
library(plotly)
library(tidyverse)
library(ggrepel)
library(gganimate)
library(tseries)
library(forecast)
```

#Line plot of all companies stocks:
```{r pressure, warning = false,echo=FALSE}

stock_plot <- ggplot(
  final_df,
  aes(Date, stock_val, group = company_name, color = factor(company_name))
  ) +
  geom_line(size=0.3) +
  scale_color_viridis_d() +
  labs(x = "Date", y = "Stock Value") +
  theme(legend.position = "top")+
  theme_minimal(base_size = 8)+ theme(axis.text.x = element_text(angle = 90))


ggplotly(stock_plot + scale_color_discrete(name="Company") )

##stock_plot + geom_point() + transition_reveal(Date)

```

Stock and Sentiment Analysis Line Plot :

```{r Line Plot}
final<- final_df%>%filter(company_name == 'AAPL' & Date>='2009-01-01' & Date <
    '2009-03-01')
```
```

```

# normalize <- function(x) {
# num <- x - min(x)
# denom <- diff(range(x))
# return (num/denom)
# }

#final$stock_val<- normalize(final$stock_val)
#final$sentiment_common<-normalize(final$sentiment_common)

final_f<- final[-1,]
final_f$stock_val<- diff(final$stock_val)
final_f$stock_lag<- lag(final_f$stock_val,n=1)

p <- ggplot() +
  geom_line(data = final_f, aes(x = Date, y = stock_lag), color =" blue") +
  geom_line(data = final_f, aes(x = Date, y = sentiment_common), color = "red"
    ) +
  scale_color_viridis_d() +
  labs(x = "Date", y = "Stock",title = "Walgreens") +
  theme(legend.position = "top")+
  theme_minimal(base_size = 8)+ theme(axis.text.x = element_text(angle = 90))

ggplotly(p)

```
Companies stock value change over time using sliding bars
```{r Sliding bar plot }

plotdata <- final_df %>%
  group_by(Date) %>%
  mutate(ordering = rank(stock_val)) %>%
  ungroup()


p<-ggplot(plotdata,
        aes(ordering, group = company_name, color=factor(company_name),fill=
    factor(company_name), show.legend = FALSE)) +
  geom_tile(aes(y = stock_val/2,
                height = stock_val,
                width = 0.9), alpha = 0.4) +
  # text on top of bars
  geom_text(aes(y = stock_val, label = as.integer(stock_val)), hjust = -0.2) +
 # geom_text(aes(y = 0, label = country, hjust = 3)) +
  # text in x-axis (requires clip = "off" in coord_cartesian)
  geom_text(aes(y = 0, label = company_name, color="black"), hjust = 0, show.
    legend = FALSE) +
  coord_flip(clip = "off", expand = TRUE) +
  enter_fade() +
   exit_shrink() +
  coord_flip()+
```

```
  scale_color_viridis_d(name="", guide=FALSE)+
  scale_fill_viridis_d(name="",guide=FALSE)+
  ylim(0, 200) +
#  theme_tufte(14,"Avenir")+
  theme_classic() +
 # guides(color=F,fill=F)+
  labs(title = "Year: {closest_state}", y="Stock", x="Company" ) +
  theme(plot.title = element_text(hjust = 0.5, size = 24),
        axis.ticks.y = element_blank(),
        axis.text.y  = element_blank()) +
  transition_states(states = Date, transition_length = 2, state_length = 1) +
#  transition_time(year)+
  ease_aes('cubic-in-out')

animate(p, nframes = 160, fps = 20, end_pause = 20, width = 500, height = 900)
#use anim_save(filename) to save

anim_save("animation_le4.gif", animation = last_animation())

```
Slope plot for change in stock value over time

```{r Slope plot}
fc<-final_df %>% filter(Date == '2008-08-08' | Date =='2016-03-01')%>%dplyr::select(Date,stock_val,company_name)
fcs<-spread(fc, key = Date, value = stock_val)


left_label <- paste(fcs$company_name, round(fcs$`2008-08-08`),sep=", ")
right_label <- paste(fcs$company_name, round(fcs$`2016-03-01`),sep=", ")

p <- ggplot(fcs) + geom_segment(aes(x=1, xend=2, y=`2008-08-08`, yend=`2016-03-01`, color=factor(company_name)), size=.5, show.legend=F) +
                  geom_vline(xintercept=1, linetype="dashed", size=.1) +
                  geom_vline(xintercept=2, linetype="dashed", size=.1) +
                  labs(x="", y="Stock Value") +  # Axis labels
                  xlim(.5, 2.5) + ylim(0,(1.1*(max(fcs$`2008-08-08`, fcs$`2016-03-01`))))

p <- p + geom_text_repel(label=left_label, y=fcs$`2008-08-08`, x=rep(1, NROW(fcs)), hjust=0.1, size=2)
p <- p + geom_text_repel(label=right_label, y=fcs$`2016-03-01`, x=rep(2, NROW(fcs)), hjust=0.1, size=2)
p <- p + geom_text(label="Time 1 : 2008", x=1, y=1.1*(max(fcs$`2008-08-08`, fcs$`2016-03-01`)), hjust=1.2, size=5)  # title
p <- p + geom_text(label="Time 2 : 2016", x=2, y=1.1*(max(fcs$`2008-08-08`, fcs$`2016-03-01`)), hjust=-0.1, size=5)  # title

# Minify theme
p + theme(panel.background = element_blank(),
          panel.grid = element_blank(),
```

```
            axis.ticks = element_blank(),
            axis.text.x = element_blank(),
            panel.border = element_blank())

```
```

8.3 Modelling

8.3.1 *Sentiment Analysis*

```
---
title: "Sentiment Analyser"
output: html_notebook
---

```{r}
library(sentimentr)
library(dplyr)
```

```{r load data}
news_data <- read.csv("Merge_Comapny_Data.csv")
head(news_data)
```

```{r}
reddit_data <- read.csv('Combined_news_final_reddit_wknd.csv')
head(reddit_data)
```

```{r get_sentiment}
news_data = mutate(news_data, news_data_new = sentiment_by(as.character(
    snippet))$ave_sentiment)
```

```{r}
news_data
```

```{r}
d <- density(news_data$news_data_new)
plot(d, main="Dist")
polygon(d, col="red", border="blue")
```

```{r}
stock_numbers <- read.csv("combined_dataframe_djia.csv")
```

```
head(stock_numbers)
```

```{r}
stock_numbers_ave <- stock_numbers%>%mutate(average_price = (High+Low)/2)%>% select(Date,company_name,High,Low,Volume,average_price)
stock_numbers_ave
```
```{r}
# library(Hmisc)
# stock_numbers_ave$lagged <- Lag(stock_numbers_ave$average_price, +1)
# stock_numbers_ave

stock_numbers_ave <- stock_numbers_ave %>%group_by(company_name) %>%mutate( target = dplyr::lead(average_price, n = 1, default = NA))%>%ungroup()
```



```{r}
news_data
```

```{r}
levels(news_data$company_name)
```

```{r}
levels(stock_numbers_ave$company_name)
```

```{r}
company_ticker_dict = list("AAPL"="Apple",
                           "AXP"="American Express",
                           "BA"="Boeing",
                           "CAT"="Caterpillar Inc.",
                           "CSCO"="Cisco",
                           "CVX"="Chevron Corporation",
                           "DD"="Dow",
                           "DIS"="The Walt Disney Company",
                           "GS"="Goldman Sachs",
                           "HD"="The Home Depot",
                           "IBM"="IBM",
                           "INTC"="Intel",
                           "JNJ"="Johnson & Johnson",
                           "JPM"="JPMorgan Chase",
                           "KO"="Coca-Cola",
                           "MCD"="McDonald's",
                           "MMM"="3M",
                           "MRK"="Merck & co",
                           "MSFT"="Microsoft",
```

````
                                  "NKE"="Nike",
                                  "PFE"="Pfizer",
                                  "PG"="Procter & Gamble",
                                  "TRV"=" The Travelers Companies",
                                  "UNH"="United Health Group",
                                  "UTX"="United Technologies",
                                  "V"="Visa Inc.",
                                  "VZ"="Verizon",
                                  "WBA"="Walgreens Boots Alliance",
                                  "WMT"="Walmart",
                                  "XOM"="ExxonMobil" )

#stock_numbers_ave%>%mutate(company_full_name = company_ticker_dict[paste0
    ("'",company_name,"'")])

company_name_list = list()
for (i in stock_numbers_ave$company_name){
  company_name_list <- append(company_name_list,company_ticker_dict[i])
}


```

```{r}
company_names_list <- stack(company_name_list)$values
```

```{r}
stock_numbers_ave <- stock_numbers_ave%>%mutate(company_name_full = company_
    names_list)
head(stock_numbers_ave)
```

```{r}
news_data%>%filter(company_name == 'IBM')
```

```{r}
a <- left_join(stock_numbers_ave, news_data, by = c("Date" = "created_time","
    company_name_full" = "company_name"))
a
```
```{r}
test <- left_join(a, reddit_data, by = c("Date" = "Date"))
```

```{r}
test_sentiment = mutate(test, sentiment_common = sentiment_by(as.character(
    News))$ave_sentiment)
test_sentiment
```
````
````

```

```{r}
write.csv(test_sentiment,"final_sentiment_data_with_reddit.csv")
```



```{r}
#final_df <- na.omit(a)
final_df <-test_sentiment%>%mutate(sentiment_score=news_data_new)%>%select(
    Date,company_name,company_name_full ,High ,Low, Volume, sentiment_score ,
    sentiment_common, target)
```

```{r}
final_df <- read.csv('Final_data_with_reddit.csv')
```

```{r}
write.csv(final_df,'Final_data_with_reddit.csv')
```

```{r}
final_df_non <- final_df
final_df_non[is.na(final_df_non)] <- 0
```

```{r}
final_df <- na.omit(final_df)
```

```{r}
write.csv(final_df_non,'Final_data_with_reddit.csv')
write.csv(final_df,"Final_data_na_filtered")
```
```

8.3.2 *Linear Modelling*

```
---
title: "R Notebook"
output: html_notebook
---
```{r}
library(keras)
library(dplyr)
```

```{r loading_data}
stock_df <- read.csv("Final_data_with_reddit.csv")
stock_df <- stock_df[,3:11]
```
```

```
head(stock_df)
```
```


```{r}
stock_df <- stock_df%>%mutate(stock_val = (High+Low)/2)
```


```{r}
stock_df <- stock_df%>%select(Date,company_name,company_name_full,stock_val,Volume,sentiment_score,sentiment_common,target)

stock_df
```

```{r}
stock_df$Date <- as.Date(stock_df$Date)
```

```{r}
sample_size = floor(0.8*nrow(stock_df))
set.seed(777)
train_stock_df = stock_df%>%filter(Date<'2014-01-01')
test_stock_df = stock_df%>%filter(Date>='2014-01-01')
```

```{r}
train_stock <- train_stock_df%>%select(stock_val,Volume,sentiment_score,sentiment_common)
test_stock <- test_stock_df%>%select(stock_val,Volume,sentiment_score,sentiment_common)
train_target <- train_stock_df%>%select(target)
test_target <- test_stock_df%>%select(target)
```

```{r}
train_data <- scale(train_stock)
col_means_train <- attr(train_data, "scaled:center")
col_stddevs_train <- attr(train_data, "scaled:scale")
test_data <- scale(test_stock, center = col_means_train, scale = col_stddevs_train)
```

```{r}
train_data <- as.data.frame(train_stock)
test_data <- as.data.frame(train_stock)
```

```{r}

```
```

```
plotting_df <- stock_df%>%select(Date,stock_val,Volume,sentiment_score,
    sentiment_common,target)
```

```{r}
plotting_df%>%ggplot() +
  geom_point(aes(x = stock_val, y = target), color = "red",alpha = 0.5)
```

```{r}
plotting_df%>%ggplot() +
  geom_point(aes(x = sentiment_score, y = target), color = "red",alpha = 0.5)
```

```{r}
plotting_df%>%ggplot() +
  geom_point(aes(x = log(1/sentiment_common), y = target), color = "red",alpha
     = 0.5)
```



```{r}
plotting_df%>%ggplot() +
  geom_point(aes(x = log(1/Volume), y = target), color = "red",alpha = 0.5)

```



```{r}
plot(density(plotting_df$target), main="Density Plot: Stock Prices", ylab="
    Frequency", sub=paste("Skewness:", round(e1071::skewness(plotting_df$
    target), 2)))

```



```{r}
sample_size = floor(0.8*nrow(stock_df))
set.seed(777)
train_stock_df = plotting_df%>%filter(Date<'2014-01-01')
test_stock_df = plotting_df%>%filter(Date>='2014-01-01')
```

```


```{r}
linear_model <- lm(target ~ stock_val+log(1/Volume)+sentiment_score+sentiment_common, data=train_stock_df)  # build linear regression model on full data
print(linear_model)
```



```{r}
summary(linear_model)

```


From the linear modelling we see that the most important factors are stock_val, Volume, and the sentiment_score of the company news. The common news data is nto considered to be incredibly informative to the Data.

```{r}
predicted <- predict(linear_model,test_stock_df)
actuals_preds <- data.frame(cbind(actuals=test_stock_df$target, predicteds=predicted))
actuals_preds
```

```{r}
correlation_accuracy <- cor(actuals_preds)

```

```{r}
mae(actuals_preds_svr$actual,actuals_preds_svr$predicteds)
```

```{r}
library(class)
library(LiblineaR)
library(e1071)
SVR_model <- svm(target ~ stock_val+log(1/Volume)+sentiment_score+sentiment_common, data=train_stock_df)  # build linear regression model on full data
print(SVR_model)
```

```{r}
pred_svr <- predict(SVR_model,test_stock_df)
actuals_preds_svr <- data.frame(cbind(actuals=test_stock_df$target, predicteds=pred_svr))
actuals_preds_svr
```
```

```
```
```{r}
correlation_accuracy_svr <- cor(actuals_preds_svr)

```
```

8.3.3 *Neural Network*

```
---
title: "R Notebook"
output: pdf_output
---


```{r loading_data}
stock_df <- read.csv("Final_data_with_reddit.csv")
stock_df <- stock_df[,3:11]
head(stock_df)
```


```{r}
x <- stock_df$sentiment_score
h<-hist(x, breaks=2, col="red", xlab="Sentiment",
    main="Histogram with Normal Curve")
xfit<-seq(min(x),max(x),length=40)
yfit<-dnorm(xfit,mean=mean(x),sd=sd(x))
yfit <- yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col="blue", lwd=2)
```

```{r}
stock_df <- read.csv("Final_data_na_filtered")
stock_df <- stock_df[,3:11]
head(stock_df)
```


```{r}
x <- stock_df$sentiment_score
h<-hist(x, breaks=10, col="red", xlab="Miles Per Gallon",
    main="Histogram with Normal Curve")
xfit<-seq(min(x),max(x),length=40)
yfit<-dnorm(xfit,mean=mean(x),sd=sd(x))
yfit <- yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col="blue", lwd=2)
```


```{r}
```

````
library(corrplot)
M <- cor(stock_df[,4:8])
corrplot(M, method="circle")
```

```{r}
stock_df <- stock_df%>%mutate(stock_val = (High+Low)/2)
```

```{r}
library(keras)
stock_df <- stock_df%>%select(Date,company_name,company_name_full,stock_val,Volume,sentiment_score,sentiment_common,target)
stock_df <- stock_df%>%filter(target!=0)
stock_df
```

```{r}
library(corrplot)
M <- cor(stock_df[,4:8])
corrplot(M, method="circle")
```

```{r}
sample_size = floor(0.8*nrow(stock_df))
set.seed(777)

# randomly split data in r
picked = sample(seq_len(nrow(stock_df)),size = sample_size)
train_stock_df =stock_df[picked,]
test_stock_df =stock_df[-picked,]

train_stock <- train_stock_df%>%select(stock_val,Volume,sentiment_score,sentiment_common)
test_stock <- test_stock_df%>%select(stock_val,Volume,sentiment_score,sentiment_common)
train_target <- train_stock_df%>%select(target)
test_target <- test_stock_df%>%select(target)
````

````
```



```{r}
train_data <- scale(train_stock)
col_means_train <- attr(train_data, "scaled:center")
col_stddevs_train <- attr(train_data, "scaled:scale")
test_data <- scale(test_stock, center = col_means_train, scale = col_stddevs_train)
```

```{r}
build_model <- function() {

  model <- keras_model_sequential() %>%
    layer_dense(units = 64, activation = "relu",
                input_shape = dim(train_data)[2]) %>%
    layer_dense(units = 64, activation = "relu") %>%
    layer_dense(units = 1)

  model %>% compile(
    loss = "mse",
    optimizer = optimizer_rmsprop(),
    metrics = list("mean_absolute_error")
  )

  model
}

model <- build_model()
model %>% summary()

```


```{r}
train_target <- data.matrix(train_target)
print_dot_callback <- callback_lambda(
  on_epoch_end = function(epoch, logs) {
    if (epoch %% 80 == 0) cat("\n")
    cat(".")
  }
)

epochs <- 300

# Fit the model and store training stats
history <- model %>% fit(
  train_data,
  train_target,
````

~~~
  epochs = epochs,
  validation_split = 0.2,
  verbose = 0,
  callbacks = list(print_dot_callback)
)

```




```{r}
library(ggplot2)

plot(history, metrics = "mean_absolute_error", smooth = FALSE) +
  coord_cartesian(ylim = c(0, 5))
```

```{r}
test_target <- data.matrix(test_target)
c(loss, mae) %<-% (model %>% evaluate(test_data, test_target, verbose = 0))

paste0("Mean absolute error on validation set: $", sprintf("%.2f", mae ))
```

```{r}
test_target <- data.matrix(test_target)
c(loss, mae) %<-% (model %>% evaluate(test_data, test_target, verbose = 0))

paste0("Mean absolute error on test set: $", sprintf("%.2f", mae ))
```

```{r}
test_predictions <- model %>% predict(test_data)
a <- as.data.frame(x = test_predictions[ , 1])
results <- a%>%mutate(actual = test_target)%>%mutate(prediction = test_predictions[ , 1]) %>% mutate(accuracy=100 - abs(((actual-prediction)/actual)*100))%>% select(actual,prediction,accuracy)
results <- results%>%filter(actual!=0)

```

```{r}
results%>%filter(accuracy<90)
```

```{r}
summary(results$accuracy)
```
~~~

8.3.4 *Recurrent Neural Network - LSTM*

~~~~
---
title: "R Notebook"
output: pdf_output
---

```{r}
library(keras)
library(dplyr)
```

```{r loading_data}
stock_df <- read.csv("Final_data_with_reddit.csv")
stock_df <- stock_df[,3:11]
head(stock_df)
```


```{r}
x <- stock_df$sentiment_score
h<-hist(x, breaks=2, col="red", xlab="Sentiment",
   main="Histogram with Normal Curve")
xfit<-seq(min(x),max(x),length=40)
yfit<-dnorm(xfit,mean=mean(x),sd=sd(x))
yfit <- yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col="blue", lwd=2)
```


```{r}
library(corrplot)
M <- cor(stock_df[,4:8])
corrplot(M, method="circle")
```

```{r}
stock_df <- stock_df%>%mutate(stock_val = (High+Low)/2)
```




```{r}
library(keras)
stock_df <- stock_df%>%select(Date,company_name,company_name_full,stock_val,
    Volume,sentiment_score,sentiment_common,target)
stock_df <- stock_df%>%filter(target!=0)
```

```{r}
~~~~

```
stock_df$Date <- as.Date(stock_df$Date)
```



```{r}
library(corrplot)
M <- cor(stock_df[,4:8])
corrplot(M, method="circle")
```



```{r}
sample_size = floor(0.8*nrow(stock_df))
set.seed(777)
train_stock_df = stock_df%>%filter(Date<'2014-01-01')
test_stock_df = stock_df%>%filter(Date>='2014-01-01')
```


```{r}
train_stock <- train_stock_df%>%select(stock_val,Volume,sentiment_score,
    sentiment_common)
test_stock <- test_stock_df%>%select(stock_val,Volume,sentiment_score,
    sentiment_common)
train_target <- train_stock_df%>%select(target)
test_target <- test_stock_df%>%select(target)
```


```{r}
train_data <- scale(train_stock)
col_means_train <- attr(train_data, "scaled:center")
col_stddevs_train <- attr(train_data, "scaled:scale")
test_data <- scale(test_stock, center = col_means_train, scale =
    col_stddevs_train)
```


```{r}
train_X <- as.matrix(train_data)
train_y <- as.matrix(train_target)
test_X <- as.matrix(test_data)
test_y <- as.matrix(test_target)


dim(train_X) <- c(dim(train_X)[1],1,dim(train_X)[2])
dim(test_X) <- c(dim(test_X)[1],1,dim(test_X)[2])
```

````
```

```{r}
library("keras")

rmsprop = optimizer_rmsprop(lr=0.00001, rho=0.9, epsilon=1e-08)
model <- keras_model_sequential()

model %>%
    layer_lstm(units            = 100,
               input_shape      = c(dim(train_X)[2],dim(train_X)[3] ),
               return_sequences = TRUE,
               ) %>%
    layer_dropout(0.5)%>%
    layer_lstm(units            = 75,
               return_sequences = FALSE,
               ) %>%
  layer_dropout(0.5)%>%
  layer_flatten()%>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dropout(0.5)%>%
  layer_dense(units = 1)

model %>%
    compile(loss = 'mae', optimizer = rmsprop)

model
```

```{r}

epochs = 50


history <- model %>% fit(
  train_X ,
  train_y ,
  epochs = epochs ,
  batch_size = 10,
  validation_split = 0.2,
  verbose = 1,
  shuffle = FALSE
)


```

```{r}
model %>% save_model_hdf5("model_lstm")

```
````
````

```
```{r}
pred_out <- model %>%
    predict(test_X, batch_size = 10) %>%
    .[,1]

length(pred_out)
```
```{r}
plot(pred_out, metrics = "mean_absolute_error", smooth = FALSE) +
  coord_cartesian(ylim = c(0, 5))

model
```

```{r}
#predicted <- as.list(pred_out)
actual <- as.list(test_y)
a <- as.data.frame(test_y)
tbl_1 <- a %>%
    mutate(predicted = pred_out)
results <- tbl_1%>% mutate(accuracy=100 - abs(((target-predicted)/target)*100)
    )%>% select(target,predicted,accuracy)
results <- results%>%filter(actual!=0)
```

```{r}
summary(results$accuracy)
```

```{r}
mae(results$target,results$predicted)
```

```{r}
output_df <- test_stock_df%>%mutate(predicted = pred_out)
```

```{r}
library(plyr)
accuracy_df <- output_df%>%mutate(accuracy=100 - abs(((target-predicted)/
    target)*100))
accuracy_df = accuracy_df%>%filter(accuracy != min(accuracy))
company_accuracy <- ddply(accuracy_df, .(company_name), summarize,
    group_accuracy=mean(accuracy))
```
```

```
```{r}
company_accuracy%>%
  ggplot() +
  geom_bar(aes(x = company_name, y = group_accuracy), color = "darkred",stat =
    "identity")+
  xlab('Company Name') +
  ylab('Mean Accuracy')+ theme(axis.text.x = element_text(angle = 90, hjust =
    1))
```

```{r}
IBM <- output_df%>%filter(company_name=='IBM')
Apple <- output_df%>%filter(company_name=='AAPL')
Walgreen <- output_df%>%filter(company_name=='WBA')
```


```{r}
library(ggplot2)
library(dplyr)
library(plotly)
library(hrbrthemes)
output_df$Date <- as.Date(output_df$Date)

# Usual area chart
p <- Walgreen %>%
  ggplot() +
  geom_line(aes(x = Date, y = target), color = "darkred")+
  geom_line(aes(x = Date, y = predicted), color="steelblue", linetype="twodash
    ") +
  xlab('Dates') +
  ylab('Stock Price-Walgreen')
p


```

```{r}
library(ggplot2)
library(dplyr)
library(plotly)
library(hrbrthemes)
output_df$Date <- as.Date(output_df$Date)

# Usual area chart
p <- IBM %>%
  ggplot() +
  geom_line(aes(x = Date, y = target), color = "darkred")+
  geom_line(aes(x = Date, y = predicted), color="steelblue", linetype="twodash
    ") +
  xlab('Dates') +
```
```

```
  ylab('Stock Price-IBM')
p

```

```{r}
library(ggplot2)
library(dplyr)
library(plotly)
library(hrbrthemes)
output_df$Date <- as.Date(output_df$Date)

# Usual area chart
p <- Apple %>%
  ggplot() +
  geom_line(aes(x = Date, y = target), color = "darkred",alpha = 0.5)+
  geom_line(aes(x = Date, y = predicted), color="steelblue", linetype="twodash
    ") +
  xlab('Dates') +
  ylab('Stock Price-Apple')
p
```

```{r}
library(ggplot2)
library(dplyr)
library(plotly)
library(hrbrthemes)
output_df$Date <- as.Date(output_df$Date)

png(filename="stock_plot_IBM.png", width=2500, height=500)
# Usual area chart
p <- IBM %>%
  ggplot() +
  geom_line(aes(x = Date, y = target,group = ), color = "red",alpha = 0.5)+
  geom_line(aes(x = Date, y = predicted), color = "black") +
  xlab('Dates') +
  ylab('Stock Price')
p

ggsave(file="stock_plot_IBM.png", width=25, height=10, dpi=300)
```

```{r}
q <- ggplotly(p)
q

```
```
```

REFERENCES

@bookbook, title=Computer architecture: a quantitative approach, author=Hennessy, John L and Patterson, David A, year=2011, publisher=Elsevier